(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 267 974**
A1

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **86115749.3**

(51) Int. Cl.[4]: **G06F 13/12 , G06F 13/28**

(22) Date of filing: **14.11.86**

(43) Date of publication of application:
25.05.88 Bulletin 88/21

(84) Designated Contracting States:
CH DE ES FR GB IT LI SE

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Dutke, Karl-Heinz, Dipl.-Phys.**
Silcherstrasse 220
D-7269 Deckenpfronn(DE)
Inventor: **Gould, J.**
5 Stofford Road Suite 8
Alliston, MA 02134(US)
Inventor: **Perchik, J.**
295 Harvard Street Apt. 607
Cambridge, MA 02139(US)

(74) Representative: **Jost, Ottokarl, Dipl.-Ing.**
Schönaicher Strasse 220
D-7030 Böblingen(DE)

(54) Control interface for transferring data between a data processing unit and input/output devices.

(57) The IBM 370 Input/Output (I/O) architecture separates I/O devices very strictly from the central processing unit and its main memory. The I/O devices can access the memory only through the channel which takes care that only accesses to authorized spaces are performed.

The invention provides a control interface that allows the I/O devices (9) to communicate with the memory directly. The memory space available for data transfer is provided to the I/O devices by two continuously running transfer mechanisms, one for inbound and one for outbound data transfer traffic. These mechanisms provide a series of buffers (2,3) which are controlled by pointers (6,7) indicating which buffers are full and which ones are empty. The current pointer values are exchanged between an application (4) and an I/O device by a third continuously running mechanism. Pointers changed are recognized by the application or by the I/O device control. The software can get control after a data transfer operation has been completed by using an interrupt mechanism. The interrupt mechanism can be tuned by the application by specifying threshold values and timers.

EP 0 267 974 A1

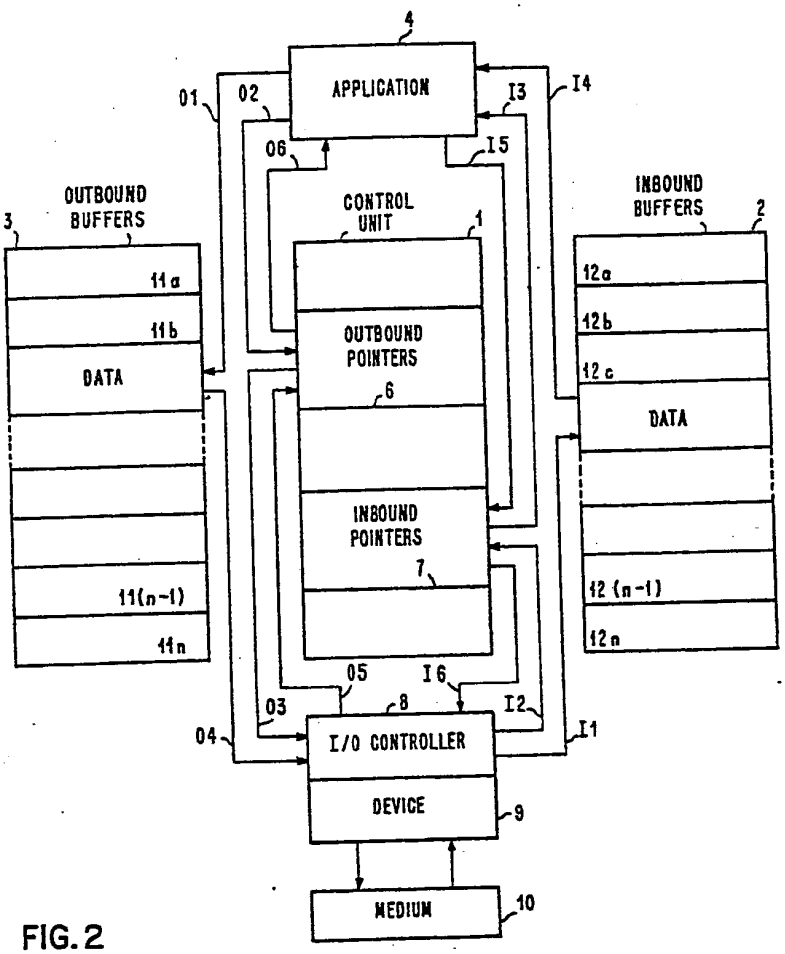
4
APPLICATION
01
02
03
04
05
06
I1
I2
I3
I4
I5
I6
CONTROL UNIT
1
OUTBOUND BUFFERS
3
11a
11b
DATA
11(n-1)
11n
OUTBOUND POINTERS
6
INBOUND POINTERS
7
8
I/O CONTROLLER
DEVICE
9
MEDIUM
10
INBOUND BUFFERS
2
12a
12b
12c
DATA
12(n-1)
12n

FIG. 2

Control Interface for Transferring Data Between a Data Processing Unit and Input/Output Devices

The invention relates to a control interface for transferring data between a data processing unit architectured on the rules of the IBM System /370 (S/370) input/output architecture, and one or more input/output devices connected thereto.

Input/output (I/O) operations based on the IBM S/370 I/O architecture involve the transfer of information between the main storage of the central processing unit (CPU) and an I/O device which is connected to the CPU. I/O devices and their control units are attached to channels which control the information (data) transfer.

The channels direct the flow of information between the I/O devices and the main storage. They relieve the CPU of the task of communicating directly with the I/O devices and permit data processing to proceed concurrently with the I/O processing.

I/O devices are attached to the channel through control units. Their operation is controlled by a control unit. From a programming point of view most control unit functions are merged with I/O device functions.

I/O activities are initiated through IBM S/370 I/O instructions (Start I/O). Specific device activities are controlled by channel programs which are initiated via an I/O instruction. The channel which can be regarded as a special CPU executes these programs in parallel to the S/370 instruction processing. At the end of the channel program an interrupt is generated which provides information on the result of the executed program. The channel programs consist of channel command words (CCW's). The command part of the CCW is communicated from the channel to the control unit which performs the specific action. If data has to be transferred for the active CCW the control unit/device provides the data or demands them. The access to the storage for fetching or storing the data is fully controlled by the channel. The devices never see any S/370 storage addresses.

The CCW's allow data traffic in only one direction: Either inbound or outbound; so with one device either inbound or outbound traffic can be active at any point in time only.

In summary an IBM S/370 I/O operation includes a starting instruction (SIO), the execution of the associated channel program controlled by the channel together with the control unit and the device, and at least one ending interrupt when the channel program is completed.

I/O architectures very widely used in microprocessor environments are totally different. There exists a very close relationship between the I/O devices and the main memory of the CPU which allows the device to access the memory directly. The I/O device itself holds the data address and initiates data transfer between itself and the memory by using these addresses. This mechanism is widely known as direct memory access (DMA). It is a very efficient concept insofar as it minimizes interference with the CPU for executing I/O operations. Once the I/O device has the addresses it can initiate data transfers whenever required without needing further excitations or requests from the processing unit. So whereas the IBM /370 channel architecture provides for optimal protection of the memory the DMA concept is a very effective means for handling data transfer.

It is thus an object of the invention to provide an interface control which enables an IBM S/370 computer system to execute a continuously running data transfer in either directions, inbound and outbound, simultaneously.

This object of the invention is accomplished by the features of the main claim. Further advantageous embodiments and implementations are disclosed in the subclaims.

Generally speaking the control interface of the invention transposes the DMA concept into the IBM S/370 I/O architecture world. The memory space available for data transfer is provided to the I/O device by two continuously running control mechanisms, one for inbound and one for outbound traffic. These control mechanisms provide a series of buffers as specified by I/O control words (CCW's). Which buffers are actually available for data traffic is determined by pointers (indicator means). The buffers are implemented as a circular buffer pool and the pointers indicate which of the buffers are full and which are empty. The current pointer values are exchanged between the application which had called for a data transfer, and the device by means of a third continuouly running control mechanism. The pointers are changed and the changes are recognized by the I/O device by means of periodically sensing the exchanged control information. This mechanism can either sense the appropriate control information or it can use an interrupt mechanism for getting control after data transfer has been completed by the I/O device. The interrupt mechanism can be tuned by the application by specifying threshold values and setting timers. Three unshared I/O control mechanisms are used for these functions: Two for the inbound and outbound data port and one for the controlling port; a fourth I/O control mechanism is used to present interruptions if required.

By applying the features of the invention to an interface control unit the following advantages will be

achieved:

Continuously running data transfers at any time in both directions at the same time controlled by the interface control unit of the invention attached to the IBM S/370 channel.

Minimization of required I/O instructions and I/O interrupts, which leads to reductions in software path lengths controlling the I/O operations, and minimization of response times between the application performing the data transfer and the I/O device.

Minimization of software turn-around times by continuously running data traffic.

Direct control of the data traffic by the application program layers without requiring system interactions, as in the conventional IBM S/370 I/O.

A complete understanding of the present invention may be obtained by reference to the accompanying drawings, when taken in conjunction with the detailed description thereof and in which:

Fig. 1 is a block diagram illustrating the components of the control interface;

Fig. 2 represents a block diagram of the various data and control lines interconnecting the components of Fig. 1;

Fig. 3 is a block diagram illustrating the control interface when used for outbound data transfers;

Fig. 4 is a block diagram of the control interface when used for inbound data transfers;

Fig. 5 is a block diagram of an interrupt control unit and

Fig. 6a, b, c are signal diagrams with respect to time showing various control signals.

Figure 1 gives a survey over the essential parts of the control interface. There are two buffer pools, 2 and 3. Buffer pool 2 is used for data transfers from an input/output device 9 to an application 4, in the following called inbound buffers. The buffers 12a - 12n of the pool are ring-connected and addressed by normal address means 5 known from the art. The other buffer pool 3 is used for outbound data transfers from an application 4 to an input/output device 9, in the following called I/O device. The outbound buffers 11a - 11n are also ring-connected. Normal address means 5 are used to access the single buffers for loading and unloading the buffers of the outbound buffer pool.

During operation the housekeeping operations for controlling the loading and unloading of the buffers is done by means of a control unit 1 which comprises two sets of pointers, the outbound pointers 6 and the inbound pointers 7, each consisting of an index pointer 6a, 7a and an acknowledgment pointer 6b, 7b. The index pointer 6a and the acknowledgment pointer 6b represent the outbound pointers 6 and the index pointer 7a, and the acknowledgment pointer 7b represents the inbound pointers 7. The inbound pointers are used to indicate and control the states of the inbound buffers 2, the states of which can be either empty (free) or full. The outbound pointers 6 are used to indicate and control the states of the outbound buffers 3.

The pointers in the preferred embodiment of the invention are one byte values allowing a representation of 256 values. Arithmetic operations performed with these values are done modulo 256, i.e. the update of the pointers, adding numbers of buffers loaded or unloaded and the comparison of the values. This modulo 256 arithmetic is used independently of the actual number of buffers provided by the application in the buffer pool. So a pointer value has no fixed relation to a specific buffer, e.g. the first buffer can be identified by a pointer number 0, 5, 10 etc. if 5 buffers are used (MAX BUFFER = 5).

The I/O controller 8 does not need to know how many buffers are really used. It uses the next loaded or unloaded buffer as long as [INDEX] - [ACKN] > 0. The application has to make sure that only the really allocated number of buffers is used by never allowing [INDEX] - [ACKN] > [MAX BUFFER].

This scheme has the advantage that all buffers can be used as long as less than 256 buffers are used. In a scheme relating the pointer values to the real buffer numbers one buffer can not be used, e.g. not be loaded for outbound data traffic, because it must be clearly defined that [INDEX] = [ACKN] means all buffers are loaded or that all buffers are empty (there is one buffer state more than can be covered with the pointer values). In the modulo 256 scheme this ambiguity can be allowed as long as less than 256 buffers are used which is beyond all practical purpose.

Further, an interrupt control 13 is provided to give control to the application when to start data transfers again after all transfers have been completed.

Referring now to Fig. 2 the interactions of the various system components will be described in connection with data and control lines connecting the system components to each other. Continuously running data transfers can be performed in either direction: From an application 4 to an I/O device 9, connected to a medium 10 or from an I/O device to an application. The outbound data traffic is routed via data line 01 (as can also be seen in Fig. 1) from an application 4 to the outbound buffers 3 and from there

via data line 04 to an I/O controller 8, which is connected to an I/O device 9.

Inbound data traffic runs over data line I1 from an I/O controller 8 to the inbound buffer(s) 2 and from the inbound buffer(s) to an application 4 via data line I4.

For an outbound data transfer the application loads the next free buffer 11i within the outbound buffer pool 3, the address of which was generated by address means, being e.g. a part of the S/370 channel. The application 4 increases the index pointer 6a of the outbound pointers 6 by a certain value via control line O2. This new index pointer is transferred via control line 03 to the I/O controller 8. This change of the outbound index pointer 6a is detected by the I/O controller 8. As a consequence the I/O controller starts the transfer of the outbound data temporarily stored in the outbound buffer(s) 3 to the I/O controller via data line 04.

Further, the I/O controller 8 increases the acknowledgment pointer 6b of the outbound pointers 6 by the same specific amount via control line O5, and the new acknowledgment pointer 6b is transmitted via control line O6 to the application 4 which indicates that the buffer(s) used for the outbound data transfer can be re-used.

For a data transfer in the opposite direction (inbound data traffic) the I/O controller 8 transfers inbound data to the next free buffer 12i of the inbound buffer pool 2 via data line I1 and increases the acknowledgment pointer 7b of the inbound pointers 7 via control line I2. The new acknowledgment pointer 7b is transferred to the application 4 via control line I3.

The application 4 detects this change from the old value of the pointer to the new one and starts the inbound data transfer from the used inbound buffer 12i of the inbound buffers 2 to the application 4 via data line I4. The application increases the inbound pointer, in this case the index pointer and transfers the new value on line I5 to the inbound pointer, which is the index pointer 7a. This indicates that the previously used buffer is now free for being used again.

The operations roughly described above will now be explained with more detail in conncection with Figs. 3, 4, 6a and 6b. Starting again with describing outbound data transfers Fig. 3 shows in greater detail the various components of the application 4 and of the I/O controller 8. The outbound data traffic in this example will transfer data from a data source 14 to one or more of the outbound buffer(s) 3 to a data sink 24 within the I/O controller 8. The data source and data sink can be of a different nature, e.g. they can be registers or storage areas representing fields in a piece of software used for data transmission operations.

The data flow is controlled by transmission gates (G) 20 and 30. The next free outbound buffer is already selected by the address means 5 which are controlled via control line 27 by the channel of the system which is not represented in the drawings or it can be controlled by any other known suitable control means. A data transmission from the data source 14 to the next free buffer is determined by control signals T1, X0' (see Fig. 6b). Control signal X0 from which XO' is derived will be generated each time the following condition is satisfied:

$$X0 \triangleq ([INDEX] - [ACKN]) < [MAX\ BUFFER],$$

where MAX BUFFER is the number of buffers used within buffer pool 3.

This means that control signal X0 is generated when the difference between the contents of the index field 15 and the acknowledgment field 16 is smaller than the contents of the MAX-BUFFER field 17. The index field 15 holds an image (duplicate) of the index pointer 6a and the acknowledgment field 16 contains an image of the acknowledgment pointer 6b.

Since at the beginning of a data transfer operation the above condition is met because the difference [INDEX] - [ACKN] is smaller than the contents of MAX BUFFER 17 control signal XO is transferred to a transmission gate 33 which generates the derived control signal XO'. The transmission gate 33 is controlled by a control signal T1 which can be regarded as an initial start signal for an outbound data transfer (as can also be seen from Fig. 6b). The control signal X0' activates the transmission gate 20 to let the data pass from the data source to the outbound buffer(s).

The control signal X0' has two further effects:

1. The control signal XO', delayed by Δt in delay element 22 is applied to a transmission gate 18 to transfer the index pointer incremented by the value V1 by an incrementor 19 into the index field 15.

2. The delayed control signal, designated XO'' is also applied to a transmission gate 21 which transmits the incremented index value via control line O2 to the index pointer 6a of the outbound pointers 6.

At the same time the new index pointer 6a is transferred via control line O3 to an index field 25 within the I/O controller 8. The comparator 45 in the I/O controller detects that the contents of the index field 25 and of the acknowledgment field 26 which had the same value before the data transfer from the data source to the outbound buffer was started, have now different values. This results in the generation of the control

signal YO. Control signal YO is transferred to a further transmission gate 34 which generates the control signal YO'. This transmission gate is controlled by a control signal T2. The control signal YO' controls transmission gate 30. The data in the outbound buffer(s) which are transparent to the data line O4 are passed from the buffer(s) 3 to the data sink 24.

Similar to the application 4 a control signal YO" which is derived from control signal YO' by a delay element 32 having a delay of Δt has also to perform two further functions:

1. To increment the value in the acknowledgment field 26 by an amount of V2, performed by an incrementor 29 and a transmission gate 28 which in turn feeds the updated acknowledgment pointer back into the acknowledgment field 26.

2. Control signal YO" is transferred to a transmission gate 31 which transmits the new acknowledgment pointer via control line O5 to the acknowledgment pointer 6b of the outbound pointers 6.

At the same time the new acknowledgment pointer is transmitted over control line O6 to the acknowledgment field 16 within the application 4. On its way to the acknowledgment field 16 the acknowledgment pointer has to pass transmission gate 36 which only in an interrupt situation is blocked by a control signal T3 from interrupt control 13. The conditions for an interrupt situation will be discussed later in the description.

By this last operation the outbound data transfer cycle is completed.

The inbound data traffic (Fig. 4) which can be understood as a data transfer from a data source 24 within the I/O controller 8 to a data sink 14 within an application 4 is again controlled by the operation cycle of a pair of pointers, in this case the index pointer 7a and the acknowledgment pointer 7b within the inbound pointers 7. At the beginning of the transfer operation the values or contents of the pointer 7a and its duplicates are [ACKN] + [MAX BUFFER] . The values of the pointer 7b and its dupplicates however, are [ACKN] .

When a data transfer is required transmission gate 30 allows the flow of data from the data source 24 to one or more of the inbound buffers 2, the address of which is set by address means 5.

The transmission gate 30 was enabled by a control signal YI' which was transmitted from the comparator 45 via a transmission gate 34 controlled by control signal T2. The new acknowledgment pointer, increased by the value V2 will be transferred again from the acknowledgment field 26 via transmission gate 31 and control line I2 to the acknowledgment pointer 7b and via the control line I3 to the acknowledgment field 16 (see Fig. 4 and 6a).

The value change of the acknowledgment pointer will be detected within the application 4, and the control signal XI is generated by the components 15, 16, 23, 17 und 35. The transmission gate 33, controlled by control signal T1 applies a control signal XI' to the transmission gate 20 so that the data bound to data sink 14 can be transferred from the input buffer(s) 2 via data line I4 to the data sink 14. Delayed by Δt, the index value in the index field 15 is increased and transmitted via transmission gate 21 and over control line I5 to the index pointer 7a within the inbound pointers 7. The new index pointer is finally transmitted over control line I6 to the index field 25 within the I/O controller 8.

The following Table 1 gives an example of changes of the pointer values in the index pointer field 15 and acknowledgment pointer field 16 in the application 4 for an outbound data traffic where the maximum number of buffers provided is 3 ([MAX BUFFER] = 3).

(Time vector from left to right.)

TABLE 1

a)

Loading of Buffers	No Buffer full	1 Buffer full (11a)	2 Buffers full (11a,b)	3 Buffers full (11a,b,c)
[INDEX] (15)	21	22	23	24 ...
[ACKN] (16)	21	21	21	21 ...

b)

Unloading of Buffers	No Buffer empty	1 Buffer empty (11a)	2 Buffers empty (11b)	3 Buffers empty (11c)
[INDEX] (15)	24	24	24	24 ...
[ACKN] (16)	21	22	23	24 ...

c)

Loading and Unloading of Buffers	2 Buffers full (11a, b) 1 Buffer empty	1 Buffer loaded (11c) 2 Buffers unloaded (11a,b)	2 Buffers loaded (11a, b) 1 Buffer unloaded (11c)	
[INDEX] (15)	24	25	27	27
[ACKN] (16)	22	24	25	25

Example a) shows how the three outbound buffers 11a-c are loaded and how the values of the pointers in the index field 15 and acknowledgment field 16 are changed during this operation. At the beginning all three buffers are free and both pointer values equal 21. In the first transfer step buffer 11a is loaded and the value of the index pointer is increased by the value V1 = 1. The new value of the pointer in the index field 15 is 22. The value of the pointer in the acknowledgment field 16 remains unchanged. When the next free buffer 11b is loaded, the value in the index pointer field 15 is increased again by 1. Its contents is now 23. The value in the acknowledgment field 16 again remains unchanged. With the last step of this example the last buffer 11c is loaded and again the value in the index field 15 is increased by 1 resulting in the final value 24. The value in the acknowledgment field 16 remains unchanged as in the previous two steps because there were no data transfers from the outbound buffers to the I/O controller 8, and only such transfers would result in a value increase of the acknowledgment pointer in the acknowledgment field 16.

The example b) shows the unloading of the buffers 11a-c. At the beginning of the unloading operation all 3 buffers are full. The pointer values in the index field 15 and the acknowledgment field 16 have the same values as they had after completion of step 3 in the preceding example, which means that the contents of the index field 15 is 24 and the one in the acknowledgment field 16 is 21. During the first step one buffer (11 a) is unloaded, which results in an increase of the value in the acknowledgment field by an amount V2 = 1 giving a total value of 22. The value in the index field 15 remains unchanged, i.e. 24.

During the second step the next buffer (11b) is unloaded and the value in the acknowledgment field 16 is increased by 1 resulting in a total of 23. After completion of the third step by which the last buffer (11c) had been unloaded the values in the index field 15 and the acknowledgment field 16 equal 24, which means that after a full outbound data transfer cycle the index field 15 and the acknowledgment field 16 hold the same value. This is the same situation as at the beginning of an outbound data transfer cycle, with the difference that the values at the beginning and at the end differ from each other in most cases. This difference equals the number [MAX BUFFER] of buffers used at most.

Example c) illustrates a mode of operation where loading and unloading take place simultaneously. It is assumed that at the beginning two buffers (11a,b) are loaded and one buffer is empty (11c). Accordingly, the value in the index field 15 is 24 and the one in the acknowledgment field 16 is 22. During the first step the free buffer (11c) is loaded and the 2 loaded buffers (11a,b) are unloaded. This results in the pointer values 25 in the index field 15 and 24 in the acknowledgment field 16. During the next step the now free buffers (11a,b) are loaded again and the full buffer (11c) is unloaded. Accordingly, the value in the index field 15 is 26 and in the acknowledgment field 16 it is 25.

Inbound data transfer operations are carried out in a very similar way.

With respect to an application 4, the following Table 2 shows the conditions under which the control signal XO for outbound data transfers and the control signal XI for inbound data transfer operations will be generated.

TABLE 2

Application 4	Outbound	([INDEX] - [ACKN]) < [MAX BUFFER]
Pointers 6	Buffers free ⇒ XO	
Inbound	([INDEX] - [ACKN]) < [MAX BUFFER]	
Pointers 7	Buffers full ⇒ XI	

Similar to Table 2 Table 3 shows with respect to an I/O controller 8 the conditions under which the control signal YO for outbound data transfers and the control signal YI for inbound data transfer operations are generated.

TABLE 3

I/O Controller 86	[INDEX] [ACKN] → Buffer available (full)
⇒YO	
7	[INDEX] [ACKN] → Buffer available (free)
⇒YI	

In general, interrupt situations may arise when the value in the acknowledgment field 26 (7b) (Fig. 4) is getting out of synchronism with the value of the acknowledgment field 16, which is possible in cases when the application 4 is not running, e.g. when there are no requests for data transfers. For these situations an interrupt control 13 is provided to the control interface.

Fig. 5 shows a block representation of the interface control unit 13, comprising a subtractor 40, a comparator 41, a register or data field 42 for storing a threshold value, a further register or data field 43 for storing a 1-bit information on whether or not an acknowledgment pointer update is allowed and an OR gate

44 generating at its output the control signal T3.

When the transmission gate 36 in Fig. 4 is blocked, the value in the acknowledgment field cannot be updated which means in other words that the acknowledgment value or acknowledgment pointer is "sleeping". The sleeping pointer SLPT is transferred via line 38 to subtractor 40. The other value, the acknowledgement pointer from the acknowledgment field 26 (7b) is transferred over line 37 to another input of subtractor 40. The difference of the two pointers is available on line 49 connected to comparator 41. The other input value of the comparator 41 is from threshold field 42 via line 46. The output of the comparator is a one bit signal which indicates, when on, that
([ACKN] - [SLPT]) ≥ [THRESHOLD]. In this case OR gate 44 generates the control signal T3. When this condition is not met, comparator 41 generates a zero bit which is transmitted on line 48 to the OR gate 44 which turns off control signal T3.

Software interrupts are initiated by the ACKN UPDATE ALLOWED field 43 which is loaded by the system software. As long as an acknowledgment pointer update is allowed the ACKN UPDATE ALLOWED field 43 contains a binary 1 bit which is transferred to OR gate 44 via line 47. This turns OR gate 44 on, irrespective of whether or not the above condition of comparator 41 is met. So control signal T3 is on again.

When a zero is loaded into the ACKN UPDATE ALLOWED field 43 OR gate 44 is turned off, when at the same time the said condition is not met. This turns the control signal T3 off.

TABLE 4

[MAX BUFFER] = 3
[THRESHOLD] = 2

		All Buffers empty (12a,b,c)	1 Buffer loaded (12a) Threshold not reached	2 Buffers loaded (12a,b), Threshold reached	
	[INDEX] (15)	33	33	33	
[SLPT] ≙	[ACKN] (16)	30	30	30	32
	[ACKN] (26)	30	31	32	

The above table 4 shows pointer values for an inbound traffic example where the ACKN UPDATE ALLOWED field 43 contains a binary 0, the value of [MAX BUFFER] is three, and [THRESHOLD] is 2.

At the beginning it is assumed that all buffers (12a,b,c) are empty. The index field 15 contains a value of 33, the acknowledgment field 16, which will contain the sleeping pointer SLPT holds a value of 30 and the acknowledgment field 26 contains 30. In the first step the first buffer (12a) will be loaded whilst the threshold is not yet reached. The index field will hold 33, the sleeping pointer 30 and the acknowledgment field 31.

During the next step two buffers (12a,b) are loaded and now the threshold is reached: the index field again remains unchanged; the same is true for the index pointer and only the value in the acknowledgment field 26 is increased by 1. Now an interrupt condition is satisfied and T3 will be unblocked. This initiates an update of ACKN (16) which in turn initiates the data transfer from the inbound buffer (2) to the data sink (14) as described before.

The treshold values may differ between inbound and outbound data transfer operations. A characteristic value for inbound data transfer operations is 1, but it can also be any random value between zero and MAX BUFFER. The characteristic threshold value for outbound data transfer operations should be MAX BUFFER -1.

Referring now to Figures 4 and 6c the previously described example will now be explained in detail. Initiated by T2 a first data transfer from the data source 24 to the first inbound buffer 12a is started. During

step 1 this first data transfer is performed. At the beginning of the next step the first data are in inbound buffer 12a.

During the second step the new acknowledgment pointer is updated and transferred to 7b. Nothing happens in the third step because T3 is in the off state. Caused by control signal T2 the second data transfer to the inbound buffer is initiated and completed at the end of step 4. At the end of step 5 all the data are in the data sink 14 and the new pointer is in the acknowledgment field 16. At the end of the sixth step the index field 25 finally holds the new index pointer and a new inbound data transfer cycle could be started.

The inbound and outbound data transfer mechanisms can run at the same time (full duplex) when the control interface components represented in Figs. 3 and 4 are provided twice in parallel.

Claims

1. Control interface for transferring data between a data processing unit and at least one input/output (I/O) device (9), with an input/output (I/O) controller (8) connecting the I/O devices to the interface, comprising

a first pool (2) of buffers (11a-n) for buffering data transferred from an I/O device to an application (4),

a second pool (3) of buffers (12a-n) for buffering data transferred from an application to an I/O device,

a control unit (1) having outbound pointer means (6) for controlling the outbound data transfer from an application to an I/O device, and inbound pointer means (7) for controlling an inbound data transfer from an I/O device to an application,

means (15-19, 21-23, 33, 35) within the application and means (25, 26, 28, 29, 31, 32, 34, 45) within the I/O controller for changing the values of the pointers and for detecting these changes for deriving inbound control signals (XI, YI) controlling the inbound, and outbound control signals (XO, YO) controlling the outbound data transfers, the control signals depending on the buffer states free (empty) or full, and

an interrupt control (13) which interrupts inbound and/or outbound data transfers if there are no free buffers available, or in cases of software interventions.

2. Control interface in accordance with Claim 1, wherein said outbound pointer means (6) and said inbound pointer means (7) each comprise an index pointer (6a, 7a) an acknowledgment pointer (6b, 7b), where the index pointers can only be changed by an application (4) and the acknowledgment pointers only by an I/O controller (8).

3. Control interface in accordance with Claims 1 and 2, wherein said changes are made by incrementing the index pointer value (INDEX) by a specific amount (V1) and the acknowledgment pointer value (ACKN) by a specific amount (V2) which can be equal to or different from the first named specific amount.

4. Control interface in accordance with one or more of the preceding claims, wherein said changes of said index pointers (6a, 7a) are performed by a mechanism (15, 18, 19, 21, 22) which is located in the application (4) and wherein the changes of said acknowledgment pointers (6b, 7b) are performed by a mechanism (26, 28, 29, 31, 32) which is located in the I/O controller (8).

5. Control interface in accordance with one or more of the preceding claims, where in said application (4) first index means (15) reflecting the value of the index pointer (6a, 7a) and first acknowledgement means (16) are provided reflecting the value of the acknowledgment pointer (6b, 7b), and where in said I/O controller (8) second index means (25) reflecting the value of the index pointer (6a, 7a) and second acknowledgement means (26) are provided reflecting the value of the acknowledgment pointer (6b, 7b).

6. Control interface in accordance with one or more of the preceding claims where in said application (4) storage means (17) are provided containing a number (MAX BUFFER) which equals the maximum number of buffers used within the first (second) pool (2, 3) of buffers.

7. Control interface in accordance with one or more of the preceding claims, where in said application (4) arithmetic and logic means (23, 35) are provided generating said inbound/outbound control signals (XI,XO) when the following condition is satisfied:

([INDEX] - [ACKN]) < [MAX BUFFER]

with [INDEX] being the value of the index pointer in said first index means (15), [ACKN] being the value of the aknowledgement pointer in said first acknowledgement means (16) and [MAX BUFFER] being the value of said maximum number of buffers in said storage means (17)

and where in the I/O controller first logic means (45) are provided generating said inbound/outbound control signals (YI, YO) when the following condition is satisfied:

[INDEX] > [ACKN]

with [INDEX] being the value of the index pointer in said second index means (25) and [ACKN] being the value of the aknowledgement pointer in said second acknowledgement means (26).

8. Control interface in accordance with one or more of the preceding claims wherein said interrupt control (13) comprises first arithmetic means (40) for forming the difference between said aknowledgement pointer (6b, 7b) and the aknowledgement pointer value in said first acknowledgement means (16) which can be regarded as a "sleeping pointer" as being not updated due to missing data transfer operations of the I/O controller, said difference being compared by second logic means (41) with a threshold value in threshold means (42) generating an interrupt control signal (T3) if the following condition is satisfied:

([ACKN]-[SLPT])≤[THRESHOLD],

said interrupt control signal being alternatively generated when an aknowledgement update allowed information contained in ackn update allowed means (43) is dropped.

9. Control interface in accordance with one or more of the preceding claims where in said application (4) a fifth transmission gate (36) is provided which is controlled by said interrupt control signal (T3) which, when blocked, does not allow an update of the aknowledgement pointer contained in said first acknowledgement means (16) by the aknowledgement pointer (6b, 7b).

10. Control interface in accordance with one or more of the preceding claims wherein data transfers between said application (4) and said inbound (outbound buffers) (2, 3) are controlled by a first transmission gate (20) within the application and wherein data transfers between said I/O controller (8) and said buffers are controlled by a second transmission gate (30) within the I/O controller.

11. Control interface in accordance with one or more of the preceding claims wherein for outbound data transfer operations the following steps are performed:

1. Dependent on said first outbound control signal (XO) the outbound data are transferred from a data source (14) within said application (4) to said outbound buffers (3) via said first transmission gate (20) and a first data line (O1),

2. the application increments the value [INDEX] of the index pointer in the first index means (15) by a specific amount (V1) and transfers the new value via a third transmission gate (21) and a first control line (O2) to the index pointer (6a) within the outbound pointers (6) and from there via a second control line (O3) to said second index means (25) within said I/O controller (8),

3. dependent on said second outbound control signal (YO) the outbound data are transfered from said outbound buffers via a second data line (O4) and said second transmission gate (30) to a data sink (24) within said I/O controller (8),

4. steps 1 to 3 are repeated until the outbound data transfers are completed.

12. Control interface in accordance with one or more of the preceding claims wherein for inbound data transfer operations the following steps are performed:

1. Dependent on said second inbound control signal (YI) the inbound data are transfered from a data source (24) within said I/O controller (8) to said inbound buffers (2) via said second transmission gate (30) and a third data line (I1),

2. the I/O controller increments the value [ACKN] of the acknowledgement pointer in the second acknowledgement means (26) by a specific amount (V2) and transfers the new value via a fourth transmission gate (31) and a third control line (I2) to the acknowledgement pointer (7b) within the inbound pointers (7) and from there via a fourth control line (I3) and said fifth transmission gate (36) to said second acknowledgement means (16) within said application (4),

3. dependent on said second inbound control signal (YI) the inbound data are transfered from said inbound buffers via a fourth data line (I4) and said first transmission gate (20) to a data sink (14) within said application (4),

4. steps 1 to 3 are repeated until the outbound data transfers are completed.

13. Control interface in accordance with one or more of the preceding claims wherein for full duplex data transfer operations the steps of the claims 11 and 12 are performed in parallel in each of two sets of said data transfer mechanisms (of Figs. 3 and 4).

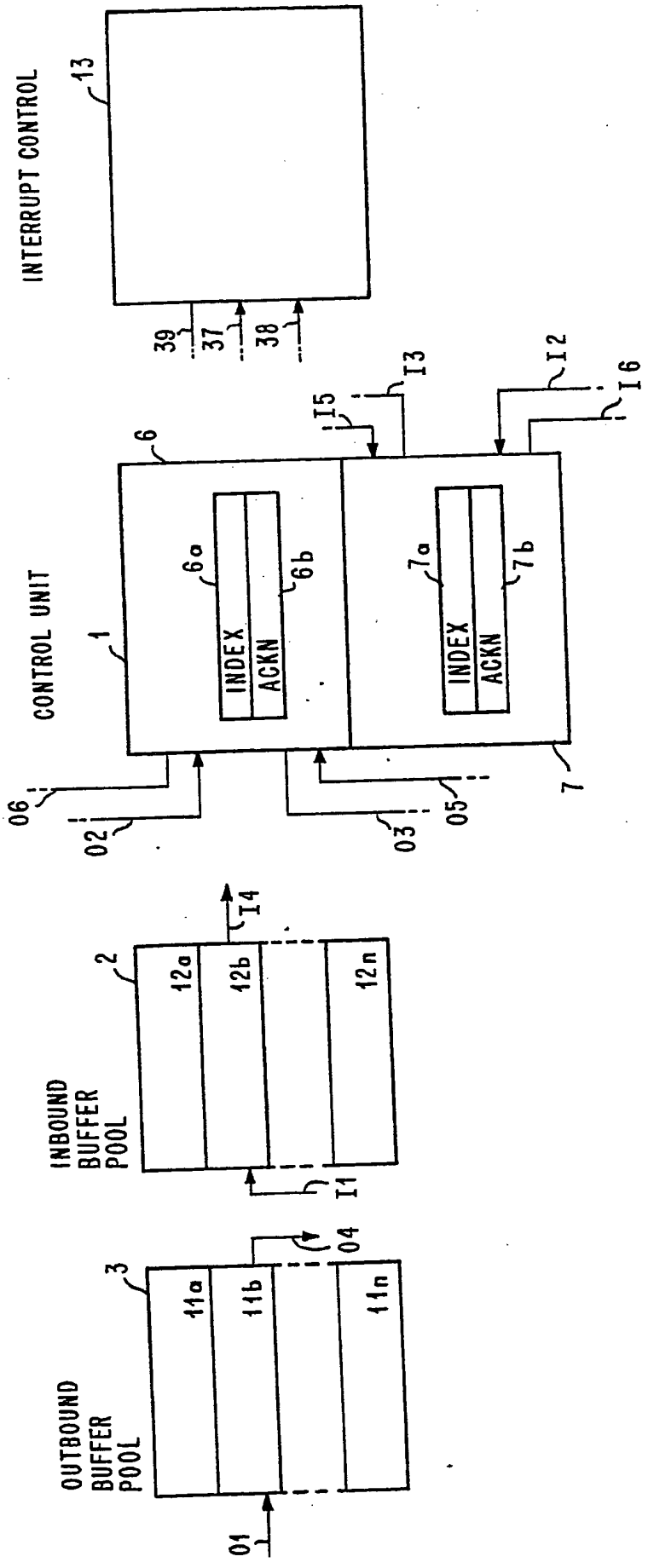
INTERRUPT CONTROL
13
39
37
38
I5
I3
I2
I6
6
CONTROL UNIT
1
6a
6b
INDEX
ACKN
7a
7b
INDEX
ACKN
06
02
03
05
7
I4
2
12a
12b
12n
INBOUND BUFFER POOL
I1
04
3
11a
11b
11n
OUTBOUND BUFFER POOL
01

FIG. 1

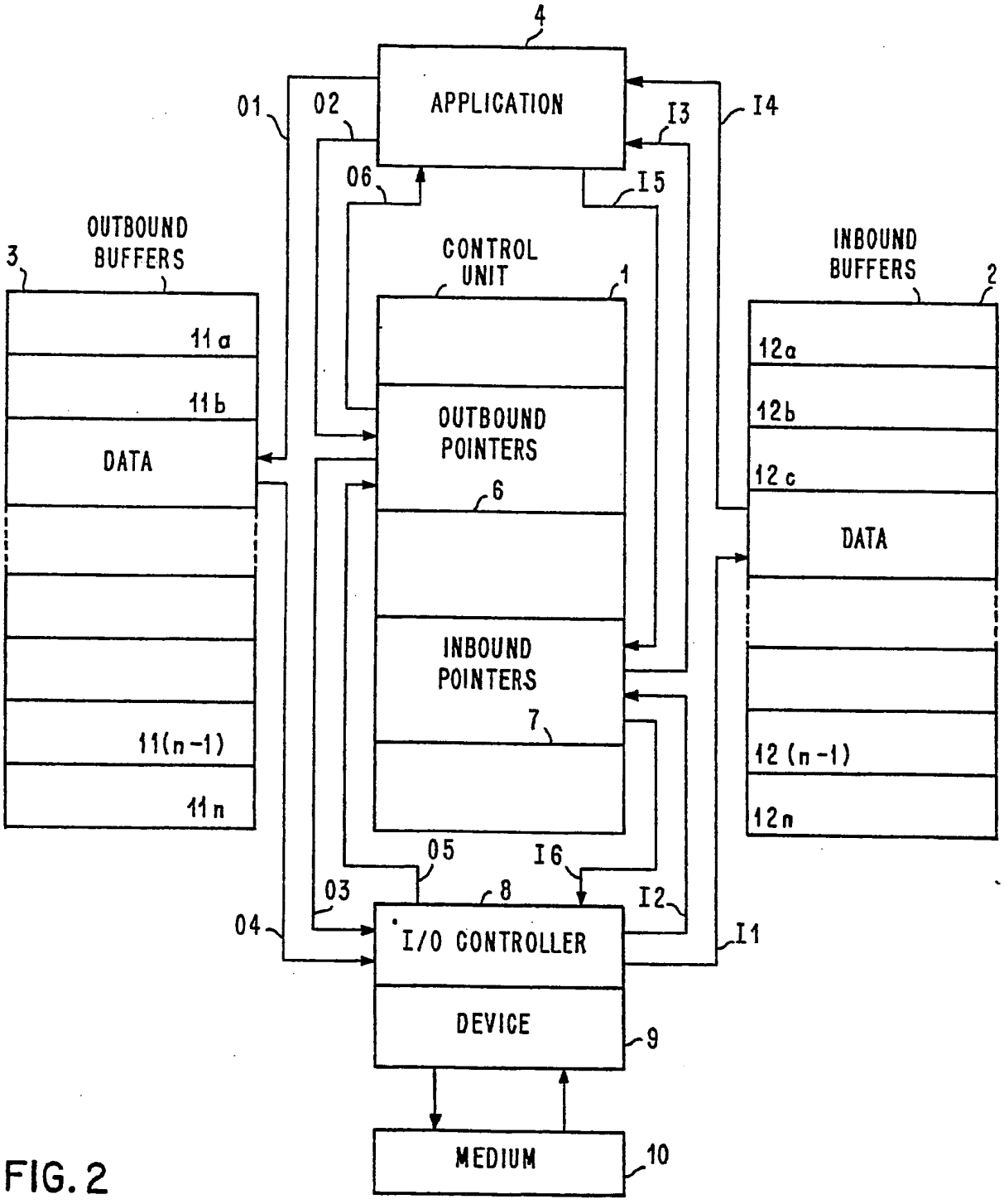
4
APPLICATION
01
02
I3
I4
06
I5
OUTBOUND BUFFERS
3
CONTROL UNIT
1
INBOUND BUFFERS
2
11a
11b
DATA
OUTBOUND POINTERS
6
12a
12b
12c
DATA
INBOUND POINTERS
7
11(n-1)
11n
12(n-1)
12n
05
I6
03
8
I2
04
I/O CONTROLLER
I1
DEVICE
9
MEDIUM
10

FIG. 2

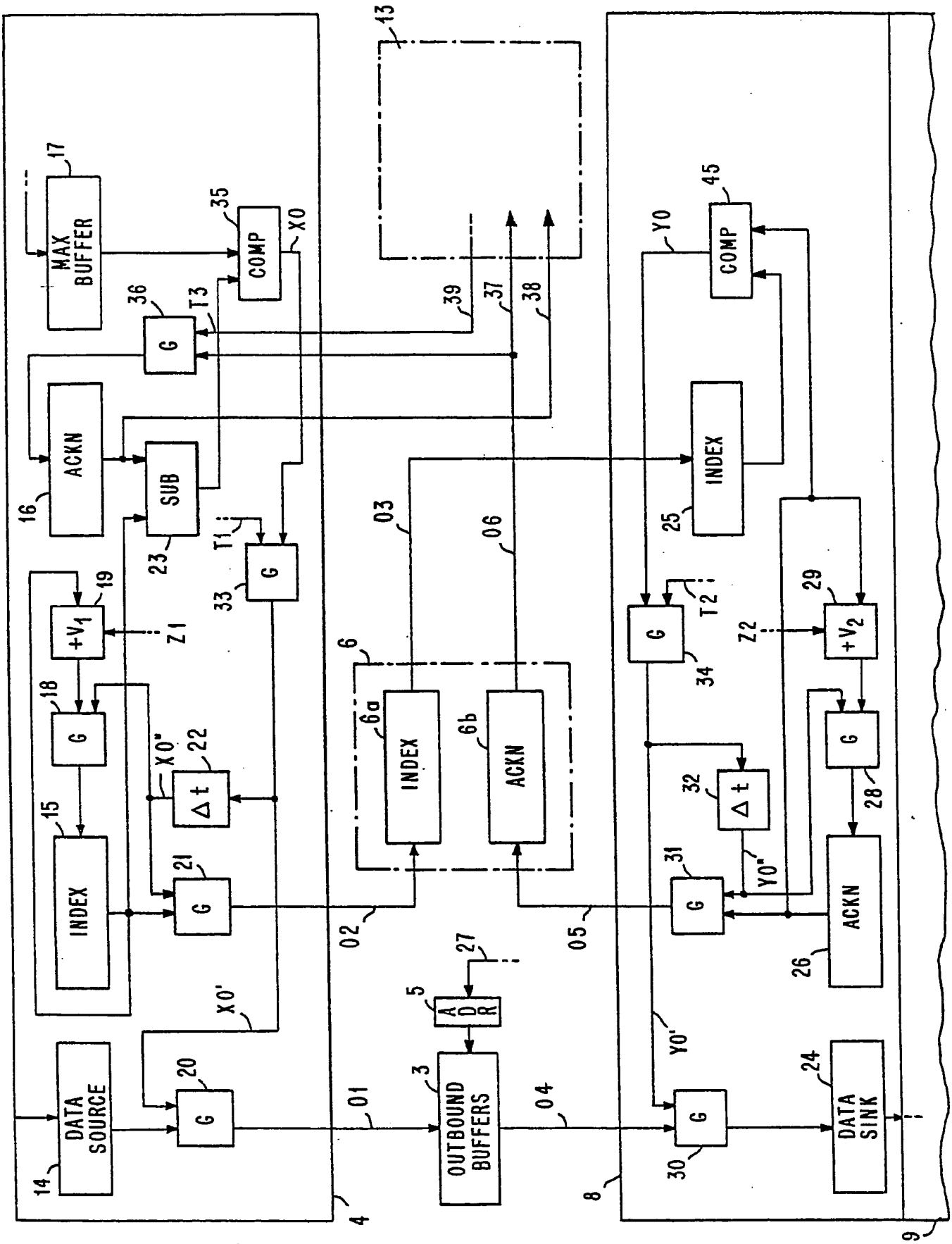
DATA SOURCE
14
G
20
X0'
01
OUTBOUND BUFFERS
3
04
DATA SINK
24
G
30
8
9
4
ADR
5
27
02
05
INDEX
6a
ACKN
6b
6
03
06
13
INDEX
15
G
21
Δt
22
X0"
G
18
+V1
19
Z1
ACKN
16
SUB
23
T1
G
33
MAX BUFFER
17
COMP
35
X0
G
36
T3
39
37
38
INDEX
25
G
34
T2
Z2
+V2
29
COMP
45
Y0
Δt
32
Y0"
G
31
Y0'
G
28
ACKN
26

FIG.3

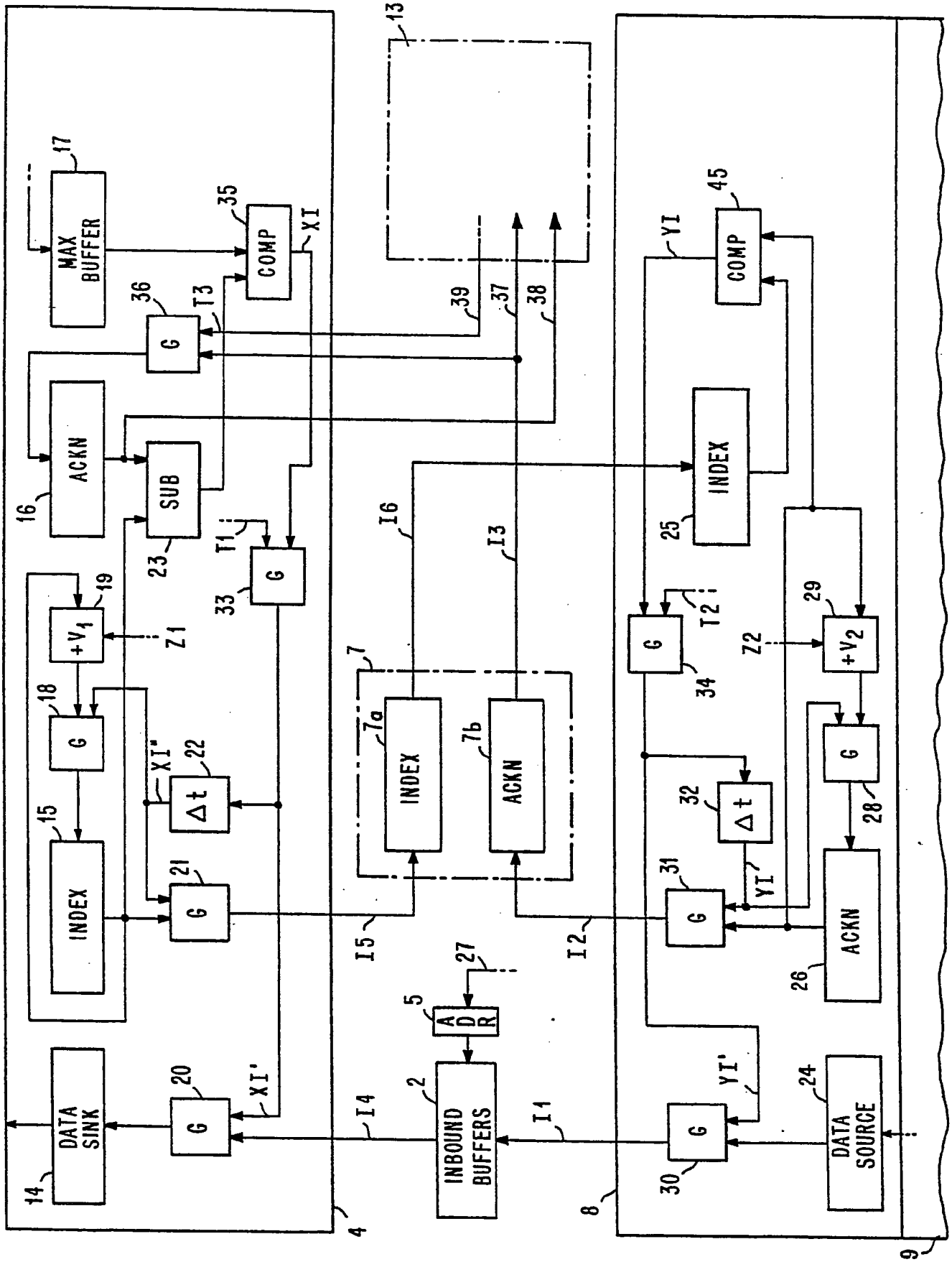
MAX BUFFER
17
COMP
35
XI
13
36
T3
39
37
38
G
ACKN
16
SUB
23
INDEX
25
45
YI
COMP
T1
I6
I3
33
G
+V1
19
Z1
7
G
T2
29
Z2
+V2
18
G
34
7a
INDEX
7b
ACKN
XI"
22
Δt
32
G
15
INDEX
21
G
31
YI'
I5
I2
27
5
ADR
26
ACKN
XI'
20
G
I4
2
INBOUND BUFFERS
I1
YI'
24
DATA SOURCE
14
DATA SINK
4
8
30
G
9

FIG.4

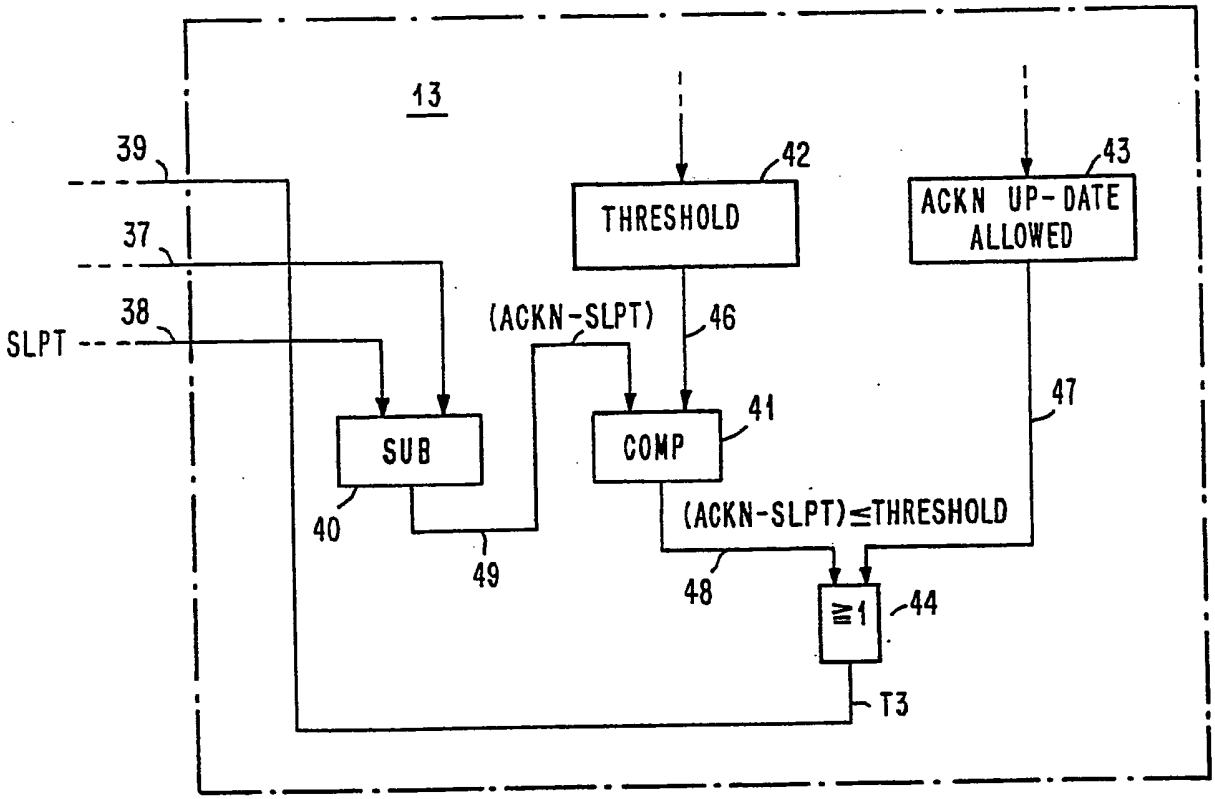
13
39
37
38
SLPT
42
THRESHOLD
43
ACKN UP-DATE ALLOWED
(ACKN-SLPT)
46
47
SUB
41
COMP
40
49
(ACKN-SLPT)≦THRESHOLD
48
≧1
44
T3

FIG.5

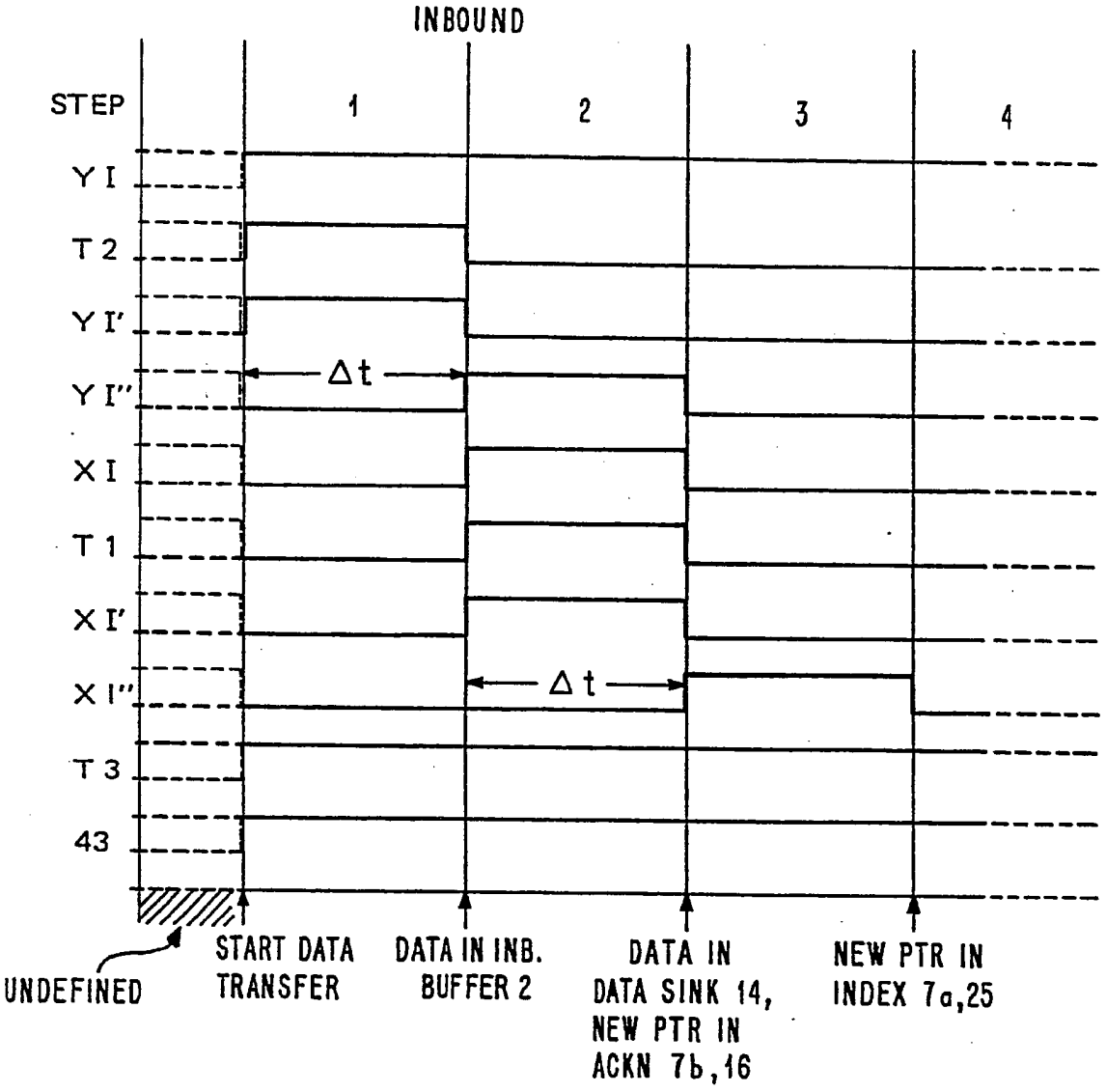
INBOUND
STEP
1
2
3
4
Y I
T 2
Y I'
Y I''
X I
T 1
X I'
X I''
T 3
43
Δt
Δt
UNDEFINED
START DATA TRANSFER
DATA IN INB. BUFFER 2
DATA IN DATA SINK 14, NEW PTR IN ACKN 7b,16
NEW PTR IN INDEX 7a,25

FIG. 6a

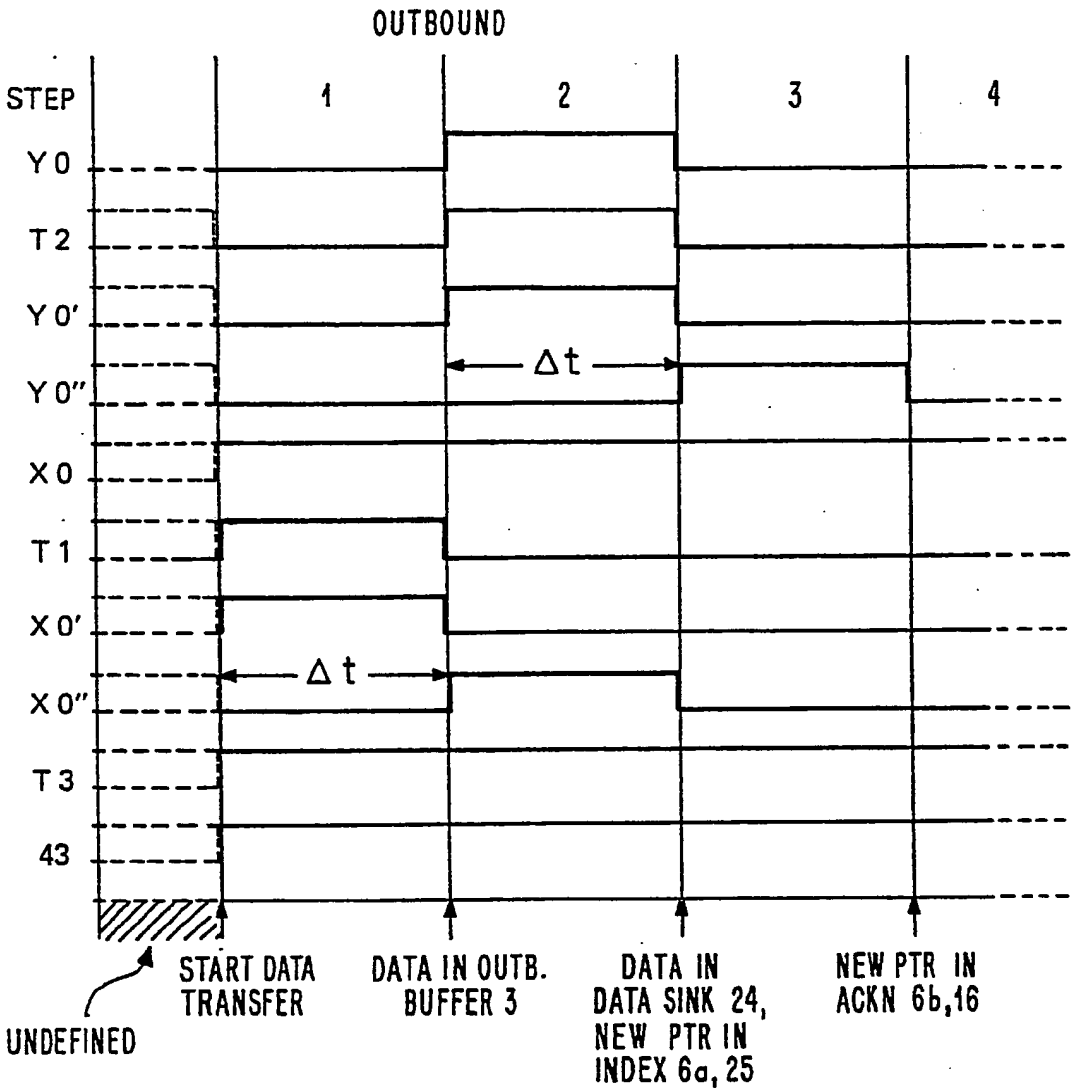
OUTBOUND
STEP
1
2
3
4
Y 0
T 2
Y 0'
Y 0"
X 0
T 1
X 0'
X 0"
T 3
43
Δt
Δt
UNDEFINED
START DATA TRANSFER
DATA IN OUTB. BUFFER 3
DATA IN DATA SINK 24, NEW PTR IN INDEX 6a, 25
NEW PTR IN ACKN 6b,16

FIG. 6 b

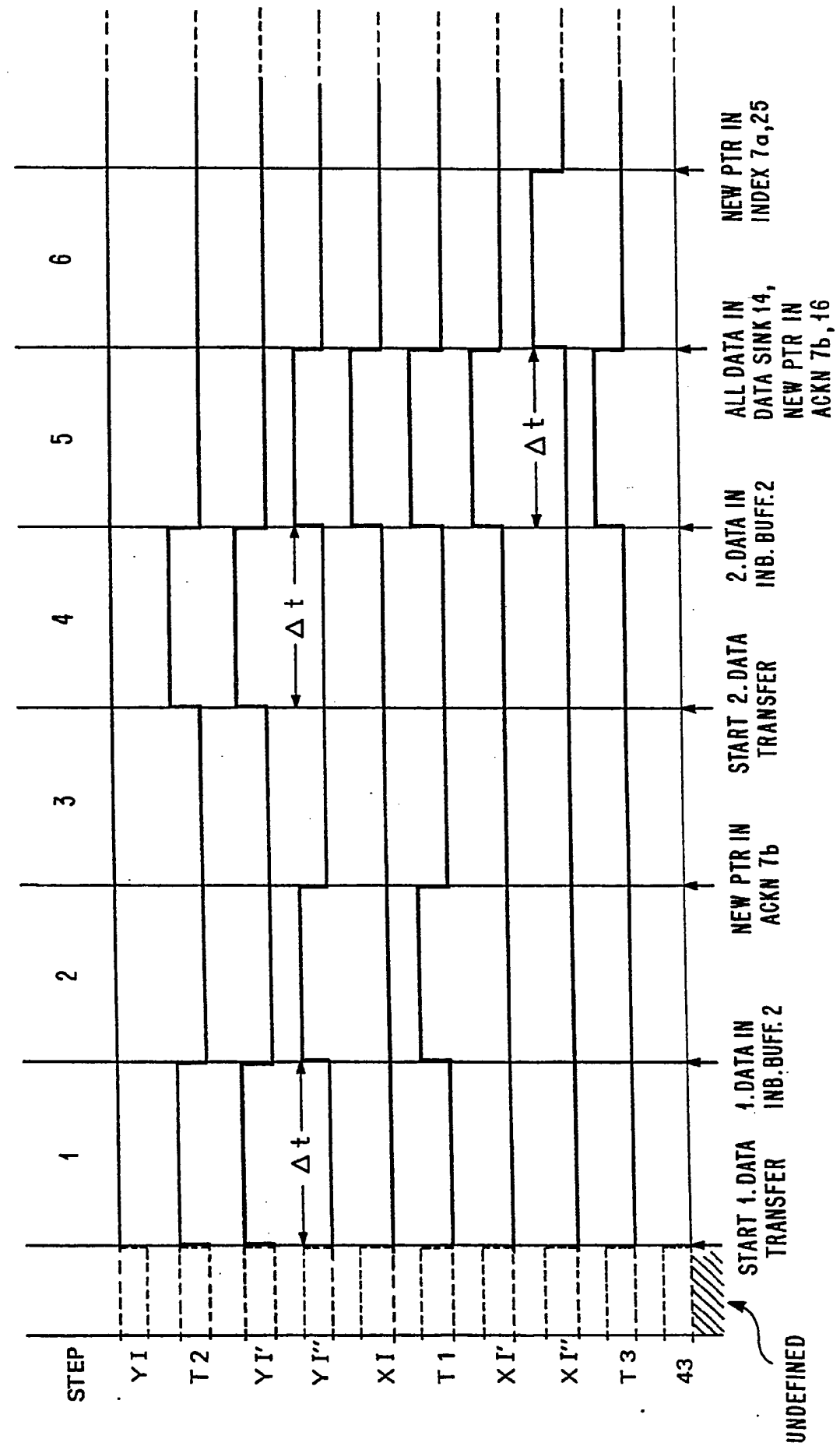
STEP
YI
T2
YI'
YI''
XI
T1
XI'
XI''
T3
43
UNDEFINED
1
2
3
4
5
6
Δt
Δt
Δt
START 1.DATA TRANSFER
1.DATA IN INB.BUFF.2
NEW PTR IN ACKN 7b
START 2.DATA TRANSFER
2.DATA IN INB.BUFF.2
ALL DATA IN DATA SINK 14, NEW PTR IN ACKN 7b,16
NEW PTR IN INDEX 7a,25

FIG. 6c

European Patent Office

EUROPEAN SEARCH REPORT

Application number

EP 86 11 5749

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.4)
X	EP-A-0 134 115 (TEKTRONIX) * Page 2, line 16 - page 5, line 20; page 9, line 4 - page 10, line 20; page 28, line 34 - page 35, line 32; figure 1 *	1	G 06 F 13/12 G 06 F 13/28

TECHNICAL FIELDS SEARCHED (Int. Cl.4)

G 06 F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	07-07-1987	WANZEELE R.J.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO Form 1503. 03.82